MS5. HPC System architecture design

Version 0.5

Documentation Information

Contract Number	2018-EU-IA-0104
Project Website	www.saintgeorgeonabike.eu
Contratual Deadline	31/05/2021
Nature	Report
Author	Maria-Cristina Marinescu (BSC)
Contributors	Artem Reshetnikov (BSC)
Reviewer	Antoine Isaac (EF), José Eduardo Cejudo Grano de Oro (EF), Cedric Bhihe (BSC), Sergio Mendoza (BSC ,Gabriela Espinosa (BSC)
Keywords	HPC, object detection, distributed implementation, batch size, OpenMPI, GPU

Co-financed by yhe Connecting Europe Facility of the European Union

Change Log

Version	Author	Description Change
V0.1	Maria-Cristina Marinescu (BSC)	First iteration
V0.2	Artem Reshetnikov (BSC)	Reviewed
V0.3	Antoine Isaac (EF), José Eduardo Cejudo Grano de Oro (EF)	Reviewed
V0.4	Ariadna Lobo (BSC)	Reviewed
V0.5	Maria-Cristina Marinescu (BSC)	Reviewed

Table of Contents

1. Introduction

This document complements the document “MS6: *System and module-level architecture development”*. It highlights foreseeable computational requirements for the implementation of techniques described in MS6. As for MS6, this document is a second iteration. An evolving implementation, growing datasets, and the potential further implementation of new techniques represent as many factors of uncertainty when evaluating the final requirements in terms of HPC architecture.

2. HPC requirements for the current system architecture and implementation

MS6 explains in detail the five main lines of work that we have been following during the first year of the project. These are as follows:

1. Improving object detection based on time context
2. Object detection based on Iconographic classes
3. Visual relationships and objects’ higher class inference based on Bounding Box analysis
4. Improving object detection based on language model
5. Caption generation based on attention mechanism
6. Caption classifier

The task that requires significant HPC resources, and which is part of all but the last line of work, is the object detector. The classifier depends only on the size of the dataset, and predicts a simple feature, that is, if a statement is likely or not to be a caption. The rest of the tasks are either relying on the MS COCO pre-trained model or on our own object detector, and caption generation does not require much computing resources.

Given the nature of Cultural Heritage (CH) as a domain we will definitely need to use our own object detection model, which is able to predict symbolic or iconographic classes, imaginary beings, symbolic actions, concepts that do not exist in everyday life as captured in pictures. At this point in the project, this is the only task that requires HPC resources, and a massive amount of them. At the same time, it is a key component which cannot be circumvented if we want to generate rich and precise metadata.

The implementation of the object detection model (both based on iconographical and technical classes) runs on the MinoTauro cluster at BSC-CNS. MinoTauro is a heterogeneous cluster in which the main computational power is provided by NVIDIA GPUs, which significantly reduces the training time of deep learning models. MinoTauro contains 39 servers, each of them contains:

- 2 Intel Xeon E5-2630 v3 (Haswell) 8-core processors, (each core at 2.4 GHz, and with 20 MB L3 cache)

- 2 K80 NVIDIA GPU Cards

- 128 GB of Main memory

- 120 GB SSD

The SGoaB project requires more resources to tune the current model and train a new object detection model based on technical classes. Experiments will be performed in parallel on two clusters. CTE-POWER is the second cluster based on IBM Power9 processors, with a Linux Operating System and an Infiniband interconnection network.

CTE-POWER has the following configuration:

- 2 login nodes and 52 compute nodes, each of them with:
    - 2 x IBM Power9 8335-GTH @ 2.4GHz (3.0GHz on turbo, 20 cores and 4 threads/core, total 160 threads per node)
    - 512GB of main memory distributed in 16 dimms x 32GB @ 2666MHz
    - 2 x SSD 1.9TB as local storage
    - 4 x GPU NVIDIA V100 (Volta) with 16GB HBM2.

The distributed implementation of the training algorithm for the object detection model uses HOROVOD - a distributed deep learning training framework for TensorFlow, Keras, PyTorch, and Apache MXNet, and MasKRCNN - a framework for instance segmentation and object detection based on the FastRCNN architecture. Underneath, the implementation is based on:

- NVIDIA CUDA: CUDA is a parallel computing platform and programming model developed by NVIDIA for general computing on graphical processing units (GPUs).

- OpenMPI: Open MPI is a framework for the Message Passing Interface implementation, which allows distributing computation in supercomputers.

Average precision is the metric of choice to understand how well our object detection model performs. Accuracy must however be balanced against computational speed. The latter does not depend only on the platform on which the model is deployed or trained. During training, two configuration metrics, namely *batch size* and *steps per epoch*, are important. *Batch size* determines the number and size of data blocks propagated through the distributed computing architecture, meanwhile *steps per epoch* denote the number of batches to be selected per epoch. Taking into account that during each epoch the model has to pass through all of the training data, the *steps per epoch* variable is calculated as train_length divided by batch_size (train_length is the size of a training dataset). This affects the amount of data transferred between the nodes of the network.

The advantages of using a batch size that is a fraction of the data set (rather than the entire dataset) are:

- Less memory is required.
  Since you train the network using fewer samples, the overall training procedure requires less memory. That's especially important if you are not able to fit the whole dataset in your machine's memory.
- Networks train faster with mini-batches.
  This is true because we update the weights after each propagation.

There are also disadvantages to using mini-batches.

- The smaller the batch, the less accurate the estimate of the gradient direction and amplitude is.
- To go through all training data, the network needs more steps per epoch. The advantages of faster training with mini-batches can be lost because the number of data transfer operations over the network increases significantly.

Considering this, scaling from a dataset with 4k images (with their bounding box information) to (a minimum of) 10k is possible with no additional resources, while maintaining the mini-batch size. The best course of action in such a case is to increase our mini-batch size to allow a corresponding gain in detection accuracy.

We conducted experiments for training the object detection model based on the Mask-RCNN architecture, using the 4k dataset mentioned above. The table below illustrates several approximations in terms of the number for batch_size, computing cores, estimated training time, and range for the dataset size, deemed sufficient for us to achieve a reasonable level of accuracy. At this point, we must also abide by execution time limitations enforced on our HPC-cluster; specifically, training cannot take longer than 48 hours, which means that we need to increase the resources in case of achieving this time limit. The model was trained with 50 epochs.

Batch size	**Resources**	**Training time**	**Size of dataset**
1000	4	1 node, 2 GPU (GPU parallelism)	13-16 hours
2000	2	1 node, 2 GPU (GPU parallelism)	48 hours
200	10	5 nodes, 10 GPU (node parallelism)	12 hours
800	10	5 nodes, 10 GPU (node parallelism)	Approximately 38-45 hours
800	16	Approximately 8 nodes, 16 GPU (node parallelism)	Approximately 38-45 hours

Nodes of the MinoTauro (2 GPUs per node) and CTE-Power (4 GPUs per node) clusters share both CPU and memory resources between our task (here the object detection model) and any number of concurrent tasks of other users. To have more computing power, we use the CTE-Power and MinoTauro clusters in parallel.

3. Discussion: Future HPC requirements

The current system implementation includes 4 techniques that all address use case 3.1 General service for enriching collections. The rest of the foreseeable use cases are:

1. Ingestion of results into Europeana (use case 3.2)
2. Search API (use case 3.3)
3. Crowdsourcing campaign for validation of enrichments (use case 3.4)
4. Uploading in data sharing platforms (use case 3.5)

Except for the first item, none of these use cases impose further computational power requirements. It is important to repeat here that the enrichment service offered by Saint George on a Bike is based on offline processing. It does not offer any real time scenario.

The search use case does not trigger any enrichment processing, but rather finds relevant information based on the metadata for the images in the existing dataset. Crowdsourcing implies running the enrichment tool for the selected collections, but the enrichment process finishes before the end user can access and validate the metadata.

In terms of global system architecture, documents MS3 and MS6 describe foreseeable options, without anticipating a final decision. This is due to many factors, among them the quality of the enrichments and the facility with which the Europeana portal and search API can be adapted to absorb and exploit annotations produced by the SGoaB project. Currently Europeana's Search APIs only exploit transcriptions, which do not include the enrichments our project would create. In any case, the decision of how to offer these services does not affect the HPC resources that the project will need.

Lastly, the implementations of new techniques or methodologies may require additional computational resources, particularly if extracting semantic graphs as metadata becomes a viable route to support enrichment. However, at this stage, we deem new computational requirements of noticeably greater magnitude improbable, given that actions, i.e. relationships between objects, will likely be deduced based on semantic information rather than intensive computer vision algorithms.